

JP6266553

Publication Title:

COMPUTER SYSTEM

Abstract:

Abstract of JP6266553

PURPOSE:To support a CISC type instruction by an RISC type processor.
CONSTITUTION:A conversion table 2 stores the correspondence between the instruction codes that are not supported by a CPU 1 and a string of instruction codes that are supported by a processor and attains a function equivalent to those instruction codes that are not supported by the CPU 1. If the instruction code supplied to the CPU 1 directly from a memory 7 or via an instruction cache 4 is not supported by the CPU 1, a controller 5 successively supplies the instruction codes forming a string of instruction codes stored in a conversion table to the CPU 1 in place of those instruction codes supplied from the memory 7.

Data supplied from the esp@cenet database - Worldwide

Courtesy of <http://v3.espacenet.com>

This Patent PDF Generated by Patent Fetcher(TM), a service of Stroke of Color, Inc.

【特許請求の範囲】

【請求項1】プロセッサと、プロセッサに供給する命令を格納したメモリとを備えた計算機システムであって、前記プロセッサがサポートしていない命令コードに対応させて、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列を記憶した変換テーブルと、

前記メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、前記メモリより供給された命令コードに代えて、当該命令コードに対応して前記変換テーブルに記憶されている命令コードの列を構成する各命令コードを前記プロセッサに順次供給する制御手段とを有することを特徴とする計算機システム。

【請求項2】請求項1記載の計算機システムであって、前記変換テーブルは、前記プロセッサより書換え可能であることを特徴とする計算機システム。

【請求項3】請求項1記載の計算機システムであって、前記変換テーブルは、プロセッサがサポートしていない命令コードによって指定されるアドレスより順次、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列と、当該命令コードの列の最終の命令コードの識別するための識別情報を記憶し、前記制御手段は、メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、当該命令コードによって指定されるアドレスより、前記識別情報で指定される当該命令コードの列の最終の命令コードまで、前記変換テーブルに記憶されている命令コードを、順次読みだして、前記プロセッサに供給することを特徴とする計算機システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、RISC (Reduced Instruction Set Computer) 型プロセッサを用いた計算機システムに関し、特に、そのエミュレーションの技術に関するものである。

【0002】

【従来の技術】近年のワークステーションにおいて用いられるMPU (Micro processor unit) は、RISC型が主流となりつつある。しかし、RISC型アーキテクチャの採用により、ハードウェア性能、機能は著しい発展をみせている反面、このようなRISC型プロセッサの性能を最大限に引き出すために必要となるコンパイラ技術等のソフトウェア技術の開発の遅れが問題となってきた (日経エレクトロニクス1990. 6. 11, P172)。

【0003】また、新たにRISC型プロセッサを採用

したコンピュータを製作する場合等、CISC型プロセッサを採用していた従来の機種との互換性の維持することが難しい (日経エレクトロニクス1992. 4. 27, P115)。

【0004】

【発明が解決しようとする課題】さて、ソフトウェアの開発工数を減らすためには、従来の蓄積されたノウハウの活用ができることが一番望ましい。特に、従来の機種のバイナリレベル (マシン語レベル) の命令セットを、そのまま新たな機種に流用可能となればより高い移植性を実現でき、また製品の互換性が保つことができる。

【0005】ところで、CISC型プロセッサは、様々な処理サイクルの複雑な命令を含む命令セットを備えている。したがって、少ないステップ数のプログラムを作成することができ、またソフトウェアの開発工数も削減できる。一方、RISC型プロセッサは複雑な命令をサポートせず、命令を同じ処理サイクルで実行できるもののみに制限することで、スムーズなパイプライン処理を実現することにより処理の高速化をはかっている。

【0006】したがって、CISC型プロセッサ上で開発したマシン語レベルのソフトウェアは、RISC型プロセッサで実行することはできない。

【0007】すなわち、CISC型プロセッサを用いたコンピュータと、RISC型プロセッサを用いたコンピュータの間には互換性がない。

【0008】このため、CISC型プロセッサを用いたコンピュータ上で開発したソフトウェア資産や、蓄積されたノウハウを、RISC型プロセッサを用いたコンピュータ上で利用することができない。

【0009】そこで、本発明は、RISC型プロセッサにおいて、そのパイプライン処理を乱すことなく、CISC型プロセッサや他のプロセッサの命令をサポートすることのできるアーキテクチャを提供することを目的とする。

【0010】

【課題を解決するための手段】前記目的達成のために、本発明は、プロセッサと、プロセッサに供給する命令を格納したメモリとを備えた計算機システムであって、前記プロセッサがサポートしていない命令コードに対応させて、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列を記憶した変換テーブルと、前記メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、前記メモリより供給された命令コードに代えて、当該命令コードに対応して前記変換テーブルに記憶されている命令コードの列を構成する各命令コードを前記プロセッサに順次供給する制御手段とを有することを特徴とする計算機システムを提供する。

【0011】

【作用】本発明に係る計算機システムによれば、たとえば、メモリよりRISC型プロセッサに供給された命令コードが、前記プロセッサがサポートしていないCISC型命令コードであった場合にも、これに代えて、当該命令コードに対応してあらかじめ、変換テーブルに記憶しておいた、当該CISC型命令と等価な機能を果たすRISC型命令コードの列を構成する各RISC型命令コードを前記RISC型プロセッサに順次供給することができる。

【0012】よって、アプリケーションプログラムの依存するプロセッサの型や種別によらずに、当該アプリケーションプログラムを実行することができる。また、この際、RISC型プロセッサに与えられる命令はRISC型命令のみであるので、そのパイプライン処理が乱れることもない。

【0013】

【実施例】以下、本発明の一実施例を説明する。

【0014】図1に、本実施例に係る計算機システムの構成を示す。

【0015】図中、1はRISC型のCPU、2は変換テーブル、3はデータキャッシュ、4は命令キャッシュ、5はコントローラ、6はシステムコントローラ、7はDRAMで構成されるメモリである。また、コントローラ5は、キャッシュコントローラ52と、エンコーダ51と、カウンタ53を内蔵した1チップコントローラICとして実現されている。

【0016】CPU1は、所望のデータをデータキャッシュ3より、所望の命令を命令キャッシュ14より取り込み処理する。所望の命令やデータが命令キャッシュ4もしくはデータキャッシュ4上に存在しない場合には、システムコントローラによってメモリ上の命令、データが、それぞれ命令キャッシュ4、データキャッシュにロードされる。各部は、システムクロック103に同期して動作する。

【0017】さて、CPU1は、システムの起動時等に、システムコントローラ6内のレジスタに、命令セットの変換を行うか否かを指定する情報をセットする。システムコントローラ6は、変換を行わない旨が指定された場合には、信号104をネグートする。信号104がネグートされると、コントローラ5のキャッシュコントローラ52は、信号100をネグートすることにより変換テーブルのバス111への出力を抑止する。一方、信号101をアサートし、命令キャッシュ2からCPU1へのバス110による直接出力をイネーブルにする。このように、命令セットの変換を行わないモードでは、変換テーブル2が無効化された状態であり、従来の計算機システムと同様に動作する。

【0018】一方、変換を行う旨が指定された場合には、システムコントローラ6は、信号104をアサート

する。信号104がアサートされると、コントローラ5のキャッシュコントローラ52は、信号100をアサートすることにより変換テーブルのバス111への出力を許可する。一方、信号101をネグートし、命令キャッシュ2からCPU1へのバス110による直接出力を抑止する。バス110による出力が抑止されると、命令キャッシュ4からCPU1への出力される命令は、バス112を介して変換テーブル2に供給される。

【0019】変換テーブル2はSRAMにより構成され、命令キャッシュとCPUの間に配置される。コントローラ5とこの変換テーブル2は、命令キャッシュから、CPU1に取り込む命令がCISC型の命令である場合には、これをRISC型の命令列に変換する。

【0020】ここで、図2に、変換テーブル2で行うCISC型命令の、RISC型命令列への変換の関係を示す。

【0021】図2のCISC型命令"ADD メモリ、レジスタ1"は、メモリの値をレジスタ1の値と加算し、元のメモリに結果を渡す加算命令である。

【0022】RISC型のプロセッサであるCPU1は、このようなメモリに対する演算命令をサポートしていない。そこで、変換テーブル2において、CISC型命令"ADD メモリ、レジスタ1"と同じ結果を得ることのできる、3つのRISC型命令よりなる命令列にCISC型命令"ADD メモリ、レジスタ1"を変換する。

【0023】すなわち、演算されるメモリの値をレジスタ2にロードする命令"LOD レジスタ2、メモリ"と、レジスタ1、2間の加算を行い結果をレジスタ1に格納する命令"ADD レジスタ1 レジスタ2"、そして、レジスタ1の内容を元のメモリにストアする命令"STA メモリ、レジスタ1"の3つのRISC型命令の命令列に、CISC型命令"ADD メモリ、レジスタ1"を変換する。

【0024】また、たとえば、図3に示したCISC型命令"ADD メモリ1 メモリ2"は、メモリ上のメモリ1というアドレスの内容に、メモリ上のメモリ1というアドレスの内容を加算する命令であるが、この命令については、4つのRISC型命令よりなる命令列"LOD レジスタ1 メモリ1"、"LOD レジスタ2 メモリ2"、"ADD レジスタ1 レジスタ2"、"STA メモリ2 レジスタ2"に変換するようにする。

【0025】このように、本実施例に係る計算機システムでは、CPU1に取り込む命令がCISC型の命令である場合には、これを同等の機能を果たすRISC型の命令列に変換する。

【0026】以下、命令コードが16ビットとして、図2で示した変換動作を例にとり、このようなCISC型命令からRISC型命令列への変換動作について、図4

を用いて説明する。

【0027】この場合、図1の変換テーブル2は、20ビット幅のアドレスを有するSRAMで構成する。

【0028】いま、CPUから命令を要求されると、コントローラ5は命令キャッシュ4もしくはシステムコントローラ8に命令フェッチを要求し、命令を一つ読みだす。読みだされた命令の各ビットは変換テーブル2を構成するSRAMのアドレスの上位16ビットに入力される。一方、コントローラ5は、カウンタ53より、SRAMのアドレスの下位4ビットとして“0h”を出力する。

【0029】今、入力された命令がCISC型加算命令1254hである場合、SRAMのアドレスは、コントローラにより生成される下位4ビット“0h”とあわせて、12540hとなる。

【0030】このアドレスには、あらかじめ、CISC型加算命令加算命令と同じ結果を得ることのできる、3つのRISC型命令よりなる命令列の第1番目のRISC型命令LOD; 1001hの下位側に、この命令列の命令数の残りを表す4ビットを連結した値を記憶させておく。この場合、残りの命令数は2個であるので、アドレス12540hには、10012hを記憶しておく。

【0031】SRAMのアドレス12540hより出力された値10012hの上位16ビット1001hは、CPU1に命令として与えられる。CPU1は、この命令を実行し、コントローラ5に次の命令を要求する。一方、SRAMのアドレス12540hより出力された値10012hの下位4ビット2hは、コントローラ5に入力される。コントローラ5は、エンコーダ51により、入力された値を判定し、入力された値が“0h”でない場合、CPUから命令要求を受け取ると、カウンタ53を1インクリメントし、SRAMのアドレスの下位4ビットを1インクリメントする。また、これによりシステムコントローラ6のウェイトの調整を行う。

【0032】これにより、SRAMのアドレスは1インクリメントされ、アドレス12541hが入力される。SRAMのこのアドレスには、あらかじめ、CISC型加算命令加算命令と同じ結果を得ることのできる、3つのRISC型命令よりなる命令列の第2番目のRISC命令ADD; 2010hの下位側に、この命令列の残りの命令数1を表す値1hを連結した値、20101hを記憶させておく。

【0033】これにより、値20101hの上位16ビットの命令ADD; 2010hは、CPU1に命令として与えられる。CPU1は、この命令を実行し、次の命令をコントローラ5に要求する。

【0034】一方、SRAMのアドレス12541hより出力された値の下位4ビットを受け取り、これが“0h”でないので、先程と同様に、CPUから命令要求を受け取ると、SRAMのアドレスの下位4ビットを1イ

ンクリメントする。

【0035】これにより、今度は、SRAMのアドレスは1インクリメントされ、アドレス12542hが入力される。SRAMのこのアドレスには、あらかじめ、CISC型加算命令加算命令と同じ結果を得ることのできる、3つのRISC型命令よりなる命令列の第3番目のRISC命令STA; 1002hの下位側に、この命令列の残りの命令数0を表す値0hを連結した値、10020hを記憶させておく。

【0036】これにより、値10020hの上位16ビットの命令STA; 1002hは、CPU1に命令として与えられる。CPU1は、この命令を実行し、次の命令をコントローラ5に要求する。

【0037】一方、コントローラ5は、SRAMのアドレス12540hより出力された値の下位4ビットを受け取り、これが“0h”であるので、カウンタ53をリセットし、SRAMのアドレスの下位4ビットを0hに戻し、次の命令フェッチを、命令キャッシュ4またはシステムコントローラ6に対し要求し、次の命令を読みだす。

【0038】さて、命令キャッシュ4またはシステムコントローラ6から入力された命令が、RISC型命令である場合には、この命令をこのまま、スルーでCPU1に渡せばよいのであるが、アーキテクチャを簡略化する為、本実施例では、SRAM内にRISC型命令に対応する値も記憶しておく。

【0039】すなわち、図5に示すように、RISC型のロード命令LOD; 1001hが入力された場合も、CISC型命令の場合と同様に、命令の各ビットをSRAMのアドレスの上位16ビットに入力し、コントローラ5は、SRAMのアドレスの下位4ビットとして“0h”を出力する。

【0040】これによって得られるSRAMのアドレス10010hには、あらかじめ、このRISC命令LOD1001hの下位側に、0h連結した値を記憶させておく。これにより、値10010hの上位16ビットの命令LOD; 1001hは、CPU1に命令として与えられる。CPU1は、この命令を実行し、次の命令をコントローラ5に要求する。

【0041】一方、コントローラ5は、SRAMのアドレス10010hより出力された値の下位4ビットを受け取り、これが“0h”であるので、SRAMのアドレスの下位4ビットを0hのままとし、次の命令フェッチを、命令キャッシュ4またはシステムコントローラ6に対し要求し、次の命令を読みだす。

【0042】なお、変換テーブル2の、CISC型命令に対応する部分の内容は、システムイニシャライズ時にCPUが設定することができる。また、システムイニシャライズ時以外のときにも、適宜に書換えることにより、様々な命令セットに対応することができるようになる。な

お、イニシャライズプログラムは、RISC型命令によって記述する。

【0043】なお、以上の実施例では、CISC型命令コードによって、直接指定されるアドレスから順次、変換テーブルに当該CISC型命令と等価な機能を果たすRISC型命令コードの列を記憶したが、これは、かならずしも、CISC型命令コードによって直接指定されるアドレスから順次記憶する必要はなく、任意のアドレスから記憶し、記憶したアドレスと対応するCISC型命令コードの対応を記憶させた適当なアドレス変換テーブルを用いて、処理を行うようにしてもよい。

【0044】以上のように、本実施例によれば、命令コードを自由に換えられるため、異なる機種間のソフトウェアの移植が容易となる。そのためソフトウェアの開発工数を削減することが可能となる。また、RISC型CPUが実行する命令は、RISC型命令のみであるので、そのパイプライン処理が乱れることもない。

【0045】

【発明の効果】以上のように、本発明によれば、RISC型プロセッサにおいて、そのパイプライン処理を乱すことなく、CISC型プロセッサや他のプロセッサの命令をサポートすることができる。

【図面の簡単な説明】

【図1】本発明の一実施例に係る計算機システムの構成を示すブロック図である。

【図2】本発明の一実施例に係る計算機システムの行う命令の変換の第1の例を示す説明図である。

【図3】本発明の一実施例に係る計算機システムの行う命令の変換の第2の例を示す説明図である。

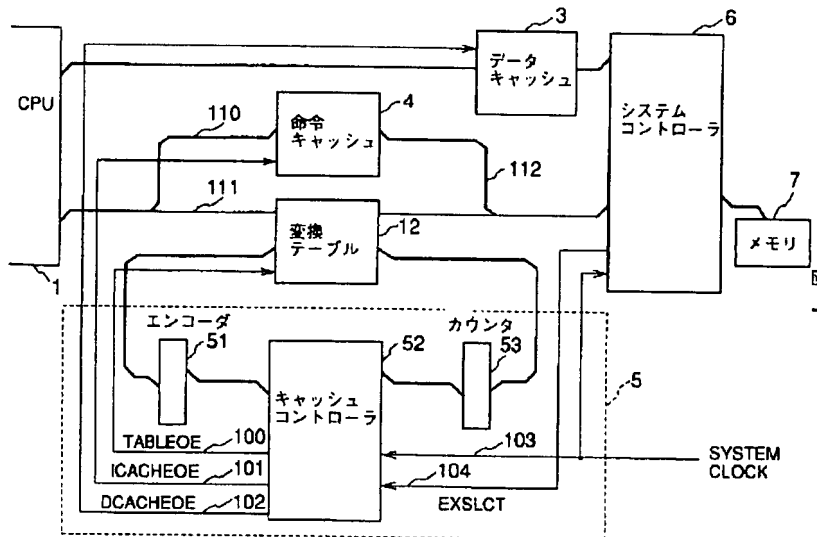
【図4】本発明の一実施例に係る計算機システムの行う命令の第1の変換動作を示す説明図である。

【図5】本発明の一実施例に係る計算機システムの行う命令の第2の変換動作を示す説明図である。

【符号の説明】

- 1 RISC型のCPU
- 2 変換テーブル
- 3 データキャッシュ
- 4 命令キャッシュ
- 5 コントローラ
- 6 システムコントローラ
- 7 メモリ
- 51 エンコーダ
- 52 キャッシュコントローラ
- 53 カウンタ

【図1】



【図 2】

【図3】

2

3

〈CISC型命令〉

〈RISC型命令〉

〈CISC型命令〉

〈RISC型命令〉

ADDメモリ,レジスタ1
(マシン語 1254h)

LODレジスタ2,メモリ
(マシン語 1001h)

ADDレジスタ1,レジスタ2
(マシン語 2010h)

STAMメモリ,レジスタ1
(マシン語 1002h)

ADDメモリ1,メモリ2

LODレジスタ1,メモリ1

LODレジスタ2,メモリ2

ADDレジスタ1,レジスタ2

STAメモリ1,レジスタ2

【図4】

【图5】

图 4

圖 5

